IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Confirmation No.: 3239
BARRETT *et al.*	Art Unit: 2183
Application No.: 10/664,055	Examiner: Aimee J. Li
Filed: September 17, 2003	Atty. Docket No.: 1875.5100000
For: **Interrupt Verification Support Mechanism**	

Second Amended Brief on Appeal to the Board of Patent Appeals and Interferences Under 37 C.F.R § 41.37

Mail Stop Appeal Briefs-Patents

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

A Notice of Appeal from the final rejection of claims 1, 2, 4-6, and 8-22 for the above-captioned U.S. Patent Application was filed on September 5, 2007, appealing the decision of the Examiner in the Final Office Action mailed April 5, 2007. Appellants hereby file one copy of this Amended Appeal Brief. The required fee set forth in 37 C.F.R. § 41.37(a)(2) was submitted with Appellants' Appeal Brief filing on May 9, 2008.

It is not believed that extensions of time are required beyond those that may otherwise be provided for in documents accompanying this paper. However, if additional extensions of time are necessary to prevent abandonment of this application, then such extensions of time are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required therefor (including fees for net addition of claims) are hereby authorized to be charged to our Deposit Account No. 19-0036.

TABLE OF CONTENTS

I. *Real Party in Interest (37 C.F.R. § 41.37(c)(1)(i))*

The real party of interest is Broadcom Corporation, having its principal place of business at 5300 California Avenue, Irvine, California, 92617. An assignment assigning all right, title, and interest in and to the patent application from the inventors to Broadcom was recorded in the U.S. Patent & Trademark Office on July 2, 2005 at Reel 016217, Frame 0328 and at Reel 016217, Frame 0314.

II. Related Appeals and Interferences (37 C.F.R. § 41.37(c)(1)(ii))

To the best of knowledge of Appellants, Appellants' legal representative, and Appellants' assignee, there are no other appeals or interferences that will directly affect or be directly affected or have a bearing on a decision by the Board of Patent Appeals and Interferences ("the Board") in the pending appeal.

III. Status of the Claims (37 C.F.R. § 41.37(c)(1)(iii))

This application was originally filed as U.S. Application No. 10/664,055 on September 17, 2003 with 24 claims. In an Amendment and Reply filed on December 14, 2006, Appellants amended claims 1, 6-18, and 21 and cancelled claim 3. Appellants re-filed the Amendment and Reply filed on December 14, 2006 in response to a Notice of Non-Compliant Amendment on January 8, 2007 with a revised listing of the pending claims with the proper status identifiers and a revised portion of the Remarks section.

The pending claims were finally rejected in an Office Action mailed April 5, 2007. A response to the April 5, 2007 final Office Action, amending claim 1 and canceling claims 7 and 23-24, was filed September 5, 2007. Claim 1 was amended to include the features of cancelled claim 7. The Advisory Action, mailed on September

20, 2007 entered the amendment, but stated that the Reply filed on September 5, 2007 did not place the application in condition for allowance. Accordingly, the claims on appeal are claims 1, 2, 4-6, and 8-22. A copy of the claims on appeal can be found in the attached Appendix as required under 37 C.F.R. § 41.37(c)(1)(viii).

IV. Status of Amendments (37 C.F.R. § 41.37(c)(1)(iv))

Subsequent to the final Office Action dated April 5, 2007, all amendments have been entered. In an Advisory Action mailed September 20, 2007 it was noted that the amendments set forth in Appellants' September 5, 2007 Amendment and Reply have also been entered.

V. Summary of Claimed Subject Matter (37 C.F.R. § 41.37(c)(1)(v))

The present invention relates to a design of a computer system that processes interrupt signals. In particular, the invention relates to an interrupt verification support mechanism to verify and to handle such interrupt signals and a method for operating the mechanism.

Overview

An interrupt signal allows a computer to suspend (i.e., interrupt) a currently executing program in order to process a higher-priority program. The interrupted program is resumed after the higher-priority program has finished executing — assuming no further interrupt signals are received. (Present Application, page 1, lines 17-25.)

In some computing systems (e.g., real-time computing systems utilized in telecommunication modules) several thousand interrupt signals per second can be received during the execution of a program. (Present Application, page 1, lines 17-

25.) Typically, however, computing systems have an associated "interrupt blocking time" that blocks the acceptance of further interrupt signals for a predetermined time interval after an interrupt signal has been received. (Present Application, page 1, lines27-30.) This "interrupt blocking time" is necessary so that the processor executing the interrupted program has time to secure its registers, react partly to the interrupt signal, and again update its registers in order to continue execution of the interrupted program. (Present Application, page 1, lines 30-33.) The longer this "interrupt blocking time" is, the more incoming interrupt signals go unprocessed. (Present Application, page 1, lines 33-34.)

It is therefore desirable to permit "the acquisition of, and reaction to the highest possible number of occurring interrupt signals per time unit." (Present Application, page 1, lines 34-36.) In order to verify the correct operation of such an interrupt processing system, it is necessary to ensure that interrupts in a large number of circumstances are correctly processed. (Present Application, page 1, line 36 - page 2, line 5.) However, "some of these circumstances are very rare and require the timing of the interrupt to be timed precisely." (Present Application, page 2, lines 5-6)

"Interrupt verification usually relies on an external interrupt generator that is part of the test bench when a design is being verified. The timing of these interrupts can be performed either uncontrolled or controlled by fixed timings. A definition of the precise point at which an interrupt strikes in a stream of instructions that are run on a processor cannot be achieved by an uncontrolled timing of interrupts, since the timing is likely to vary and it does not allow the same test to be reproduced. A timing of interrupts controlled by fixed timings might obtain a determined point of

interruption of the instruction flow, but is liable to change when the timing of internal signals changes, e.g. when a bug is fixed." (Present Application, page 2, lines 14-21.)

Therefore, the present invention provides a device and a corresponding method that are "able to define the precise point at which an interrupt strikes in a stream of instructions run on a processor" and the ability to "reliably reproduce the same test results on the silicon design as in a simulation." (Present Application, page 2, lines 27-32.)

Independent claim 1

Independent claim 1 recites a method of processing an interrupt verification support mechanism in a computer system comprising a processor and an input for external interrupts communicatively coupled to the processor, the method comprising the steps: (a) processing at least one actual instruction in the processor (Present Application, page 6, lines 19-24; FIG. 4, element 4), and (b) if an external interrupt request or an interrupt pseudo-instruction is received by the processor, comparing data content of a program counter with data content of an interrupt register and replacing the actual instruction in an instruction fetch stage of the processor with the pseudo-instruction when the data content of the program counter matches the data content of the interrupt register, or when an external interrupt is present (Present Application, page 6, lines 6-24; FIG. 4, elements 1, 7).

Independent claim 8

Independent claim 8 recites an interrupt verification support mechanism device for a computer system comprising a processor and an input for external interrupt requests or interrupt pseudo-instructions communicatively coupled to the processor (Present Application, page 6, lines 19-24), wherein the device includes a set

of one or more interrupt registers each of which contains information (Present Application, page 6, lines 6-10; FIG. 4, element 1), the information including at least a program counter of the instruction which is to be interrupted and a sort of interrupt to use, so as to enable the device to process at least one actual instruction (Present Application, page 6, lines 6-17; FIG. 4, element 1), and if an external interrupt request is received by the processor, the at least one actual instruction is replaced with the pseudo-instruction (Present Application, page 6, lines 19-24).

VI. Grounds of Rejection to be Reviewed on Appeal (37 C.F.R. § 41.37(c)(1)(vi))

In the final Office Action mailed April 5, 2007, the Examiner rejected claims 1-2, 5-8, and 23-24 under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,704,863 to Paul *et al.* ("Paul"), claims 4 and 20-22 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Paul in view of Sproul, III, U.S. Patent No. 4,498,136 ("Sproul"), and claims 9-19 under 35 U.S.C. § 103(a) as allegedly being unpatentable over Paul in view of Case *et al.*, U.S. Patent No. 4,777,587 ("Case").

In an Amendment and Reply filed September 5, 2007, in response to the final Office Action mailed April 5, 2007, Appellants amended claim 1 to include the features of dependent claim 7 and cancelled claims 7 and 23-24. The Advisory Action of September 20, 2007 indicated that the amendments made in Appellants September 5, 2007 Amendment and Reply were entered. Thus, claims 1, 2, 4-6, and 8-22 are pending in the application.

Accordingly, the grounds of rejection to be reviewed on appeal are:

A. Rejection of claims 1, 2, 5-6, and 8 under 35 U.S.C. § 102(e) over U.S. Patent No. 6,704,863 to Paul.

B. Rejection of claims 4 and 20-22 under 35 U.S.C. § 103(a) over Paul in view of U.S. Patent No. 4,498,136 to Sproul.

C. Rejection of claims 9-19 under 35 U.S.C. § 103(a) over Paul in view of U.S. Patent No. 4,777,587 to Case.

VII. Argument (37 C.F.R. § 41.37(c)(1)(vii))

In the remarks that follow, the rejection of each claim is separately discussed. Appellants recognize that any claim that depends from a patentable independent claim is patentable at least by virtue of its dependency. In addition, claims 2 and 4-6, which depend from claim 1 and claims 9-22, which depend from claim 8, are patentable not only in view of their dependencies, but also by virtue of their own respective features.

There are three separate grounds of rejection to be reviewed on appeal.

A. The rejection of claims 1, 2, 5-6, and 8 under 35 U.S.C. § 102(e) as being allegedly anticipated by Paul is improper and must be reversed

In section 4 of the final Office Action dated April 5, 2007, the Examiner rejected claims 1, 2, 5-6, and 8 and under 35 U.S.C. § 102(e) as being allegedly anticipated by Paul. Appellants respectfully traverse this rejection.

Claim 1

Independent claim 1 recites, among other features:

> if an external interrupt request or an interrupt pseudo-instruction is received by the processor, **comparing** data content of a program counter with data content of an interrupt register and replacing the actual instruction in an instruction fetch stage of the processor with the pseudo-instruction when the data content of the program counter matches the data content of the interrupt register, or when an external interrupt is present.

Paul does not teach or suggest at least this feature of independent claim 1.

Paul is directed to a processor that allegedly provides reduced latency and loss of processor bandwidth when responding to an interrupt. Paul's processor purportedly achieves this, in part, by directly inserting an interrupt's associated

instruction(s) into the pipeline of the processor; thereby avoiding "the need for the processor to save its context and branch to an interrupt service routine in memory." (Paul, col. 3, lines 20-29.) Once an interrupt is received in Paul, a determination is made as to whether the received interrupt is a low latency interrupt (inline with the invention of Paul) or a conventional interrupt. If the interrupt is a low-latency interrupt, the processor pipeline is stalled and then instruction(s) associated with the received interrupt are inserted into the processor pipeline. After the instruction(s) of the interrupt are inserted, the pipeline is restarted and the interrupted program continues execution. (Paul, col.7, lines 9-23; FIG. 3.)

In contrast to the teachings of Paul, the invention of claim 1 does not simply insert instructions associated with a received interrupt into the pipeline of a processor. Rather, in claim 1, if an external interrupt request or an interrupt pseudo-instruction is received, the data content of a program counter is **compared** with the data content of an interrupt register and "when the data content of the program counter matches the data content of the interrupt register or when an external interrupt is present," the actual instruction in an instruction fetch stage of the processor is replaced with the pseudo-instruction. The comparison operation of claim 1 is done regardless of whether an external interrupt is present or an interrupt pseudo-instruction is received. A specific implementation of this feature is shown in FIG. 4 (reproduced below) of the present application.

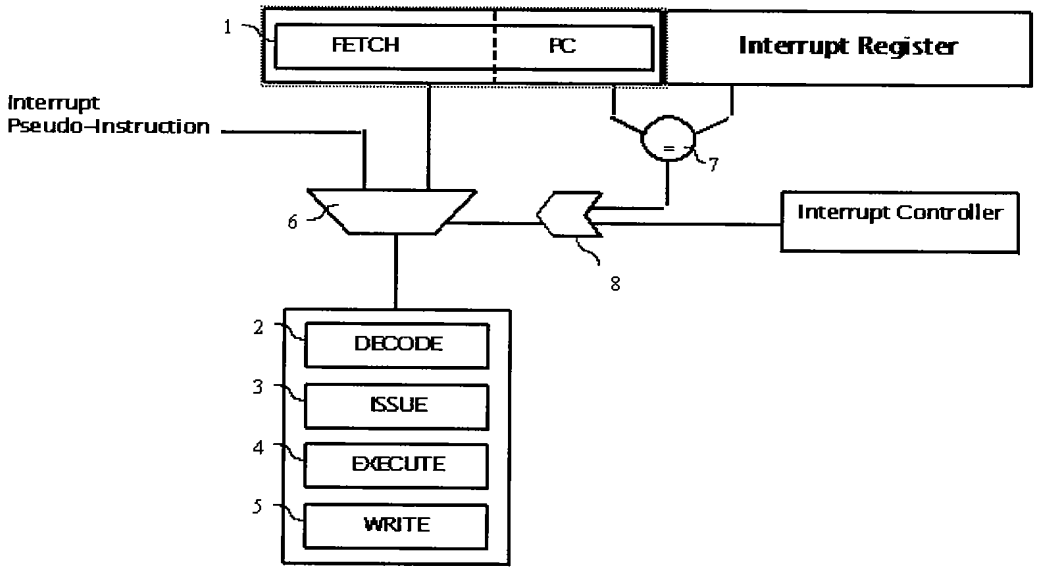

FIG. 4

Paul does not teach or suggest such a **comparison** operation as described in claim 1 and illustrated by element 7 of FIG. 4. Furthermore, the Examiner fails to specifically disclose where Paul teaches or suggests this feature. The Examiner merely states on page 6 of the final Office Action that:

> Paul has taught the method claim 1 further comprising comparing data content of a program counter with data content of an interrupt register and replacing the actual instruction with a pseudo-instruction when the data content of the program counter matches the data content of an interrupt register, or when an external interrupt is present (Paul column 7, lines 10-23 "...the pipeline is stalled 76 at the insertion point and the first of the hardwired interrupt-related instructions is inserted 78 into the pipeline..."; column 8, line 26-64 "...When the interrupt handling circuitry detects that the interrupt 104 is a BARq DMA request, it stalls the pipeline by sending a stall signal to fetch stage 86, enables clocking of the hardwire instruction stack, and programs the multiplexor 96 to connect the hardwired instruction stack to the insertion point..."; Figure 3; Figure 5; and column 9, lines 25-41 "Other types of interrupts may be handled by the processor according to this method in the conventional way (i.e., when an interrupt occurs, the processor saves the current context [program counter and status]

> and replaces the contents of the instruction pipeline with the beginning instructions of the interrupt service routine)...").

None of the sections referred to by the Examiner teach or suggest the **comparison** operation of claim 1.

Furthermore, the feature of claim 1, noted above, allows for the **precise** point at which an interrupt is to occur in a stream of instructions; not until the data content of the program counter matches the data content of the interrupt register is the pseudo-instruction inserted into the pipeline of the processor. Using the method described in claim 1, an interrupt processing system is able to be verified in a large number of circumstances, some of which require the timing of the interrupt to be timed precisely. Paul does not describe any method that can provide for the precise point at which an interrupt is to occur in a stream of instructions, let alone a **comparison** operation such as the one disclosed in claim 1 used, in part, to achieve this timing precision.

Because Paul does not teach or suggest each and every feature of claim 1, it cannot anticipate that claim. Accordingly, the rejection of claim 1 under 35 U.S.C. §102(e) is improper and must be reversed. Furthermore, dependent claims 2 and 5-6 are also not anticipated by Paul for at least the same reasons as independent claim 1 from which they depend and further in view of their own respective features. Accordingly, the Examiner's rejection of claims 2 and 5-6 must also be reversed.

Claim 8

Independent claim 8 recites, among other features:

> a set of one or more interrupt registers each of which contains information, the information including at least a program counter of the instruction which is **to be** interrupted and a sort of interrupt to use...

The Examiner wrongly alleges that Paul teaches this feature of claim 8.

In Paul, interrupt handling circuitry detects that an interrupt request occurs and subsequently stalls the pipeline. Once the pipeline is stalled, the instructions associated with the interrupt are inserted into a predetermined stage in the pipeline (e.g., decode stage). (Paul, col. 8, lines 39-44.) Nowhere does Paul contemplate the use of an interrupt register that stores "a program counter of the instruction which is **to be** interrupted", as recited in claim 8 (emphasis added). A specific implementation of this interrupt register is illustrated in FIG. 4 (reproduced above) and described at page 21, lines 10-24 of the present application:

> [w]hen the program counter PC matches the interrupt register, or an external interrupt is present, the actual instruction is replaced with a pseudo-instruction.

The value stored in the interrupt register allows for the precise point at which an interrupt is to occur in a stream of instructions. (Present Application, page 2, lines 27-29.)

The Examiner, on page 8 of the final Office Action, states that Paul teaches of two types of interrupt handling and in both methods "the instruction address of the instruction **being** interrupted is stored, which is stored in the program counter during execution of the main program" (emphasis added). However, Appellants point out that storing the instruction that is **being** interrupted is completely different to storing the instruction which is **to be** interrupted. As is well known in the relevant art(s), conventional systems store the program counter of the instruction that is **being** interrupted so as to restart execution of the interrupted program from the point of interruption (commonly referred to as saving the processor's context). The interrupt registers of claim 8 are not used in this conventional scheme. Rather, the interrupt registers of claim 8 store the program counter of the instruction which is **to be**

interrupted. As noted before, this allows for the precise point at which an interrupt is to occur in a stream of instructions. (Present Application, page 2, lines 27-29.)

Because Paul does not teach or suggest each and every feature of claim 8, it cannot anticipate that claim. Accordingly, the rejection of claim 8 under 35 U.S.C. §102(e) is improper and must be reversed.

B. The rejection of claims 4 and 20-22 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Paul in view of Sproul is improper and must be reversed

Claim 4 depends from independent claim 1 and includes the features recited therein. Sproul does not overcome all of the deficiencies Paul relative to claim 1, described above. For at least this reason, the rejection of claim 4 must be reversed.

Claims 20-22 depend from independent claim 8 and include the features recited therein. Sproul does not overcome all of the deficiencies Paul relative to claim 8, described above. For at least this reason, the rejection of claims 20-22 must be reversed.

C. The rejection of claims 9-19 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Paul in view of Case is improper and must be reversed

Claims 9-19 depend from independent claim 8 and include the features recited therein. Case does not overcome all of the deficiencies Paul relative to claim 8, described above. For at least this reason, the rejection of claim 9-19 must be reversed.

VIII. Conclusion

The subject matter of claims 1, 2, 4-6, and 8-22 are patentable over the cited art made of record. Therefore, Appellants respectfully request that the Board reverse the Examiner's final rejection of these claims under 35 U.S.C. §102 and remand this application for issue.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.

Robert Sokohl
Attorney for Appellants
Registration No. 36,013

Date: June 26, 2009

1100 New York Avenue, N.W.
Washington, D.C. 20005-3934
(202) 371-2600

997583_1.DOC

CLAIMS APPENDIX

1. *(Previously Presented)* A method of processing an interrupt verification support mechanism in a computer system comprising a processor and an input for external interrupts communicatively coupled to the processor, the method comprising the steps:

(a) processing at least one actual instruction in the processor; and

(b) if an external interrupt request or an interrupt pseudo-instruction is received by the processor, comparing data content of a program counter with data content of an interrupt register and replacing the actual instruction in an instruction fetch stage of the processor with the pseudo-instruction when the data content of the program counter matches the data content of the interrupt register, or when an external interrupt is present.

2. *(Original)* The method of claim 1 comprising :

processing at least one actual instruction in the processor in an instruction pipeline wherein instructions are processed concurrently by an instruction fetch stage, an instruction decode stage, an instruction issue stage, an instruction execute stage and a result write-back stage.

3. *(Cancelled)*

4. *(Original)* The method of claim 1 further comprising:

creating the pseudo-instruction by a co-processor connected to the processor.

5. *(Original)* The method of claim 1 comprising:

simultaneously processing a number of instructions in the processor in an

instruction pipeline with several instruction stages each instruction being in a different instruction stage at a time.

6. *(Previously Presented)* The method of claim 1 further comprising:

storing at least information of a program counter of the instruction which is to be interrupted and a sort of interrupt to use in a set of one or more interrupt registers of the processor.

7. *(Cancelled)*

8. *(Previously Presented)* An interrupt verification support mechanism device for a computer system comprising a processor and an input for external interrupt requests or interrupt pseudo-instructions communicatively coupled to the processor, wherein the device includes a set of one or more interrupt registers each of which contains information, the information including at least a program counter of the instruction which is to be interrupted and a sort of interrupt to use, so as to enable the device to process at least one actual instruction, and if an external interrupt request is received by the processor, the at least one-actual instruction is replaced with the pseudo-instruction.

9. *(Previously Presented)* The device of claim 8 wherein

the device further comprises an instruction fetch with a program counter and an interrupt register, the instruction fetch being coupled to a first input of a multiplexer for transmitting instructions to said multiplexer, a second input of the multiplexer connected

to an interrupt pseudo-instruction input and the program counter connected with the interrupt register by a comparator.

10. *(Previously Presented)* The device of claim 9 wherein

the second input of the multiplexer is capable of receiving interrupt pseudo-instruction signals or external interrupt requests.

11. *(Previously Presented)* The device of claim 9 wherein

the comparator creates a high level signal only if data content of the program counter matches data content of the interrupt register.

12. *(Previously Presented)* The device of claim 9 wherein

an output of the comparator is connected to a first input of an or-operator, and a second input of the or-operator is connected to an interrupt controller so as to enable the or-operator to create a high level signal if a signal received from the interrupt controller differs from a signal received from the comparator.

13. *(Previously Presented)* The device of claim 9 wherein,

when data content of the program counter matches data content of the interrupt register, the actual instruction is replaced with a pseudo-instruction.

14. *(Previously Presented)* The device of claim 9 wherein

when an external interrupt request is present at the multiplexer, the actual

instruction is replaced with an interrupt pseudo-instruction.

15. *(Previously Presented)* The device of claim 9 wherein

an instruction coming from an output of the multiplexer is sequentially processed in an instruction pipeline of the processor.

16. *(Previously Presented)* The device of claim 9 wherein

an instruction pipeline of the processor includes an instruction fetch stage, an instruction decode stage, an instruction issue stage, an instruction execute stage and a result write-back stage.

17. *(Previously Presented)* The device of claim 9 wherein

the interrupt pseudo-instruction effects instruction state stages required by the interrupt pseudo-instruction.

18. *(Previously Presented)* The device of claim 16 wherein

if an interrupt request or an interrupt pseudo-instruction is received by the processor, the processor is adapted to cancel an instruction that is in the instruction fetch stage when the interrupt request or the interrupt pseudo-instruction is received and to reissue the cancelled instruction starting at the instruction fetch stage.

19. *(Original)* The device of claim 16 wherein

if an interrupt request or an interrupt pseudo-instruction is received by the

processor, the processor is adapted to cancel an instruction that is in any instruction stage when the interrupt request or the interrupt pseudo-instruction is received and to reissue the instruction starting at the instruction fetch stage.

20. *(Original)* The device of claim 8 wherein

the pseudo-instruction is created by a co-processor connected to the processor.

21. *(Original)* The device of claim 20 wherein

the device is a media decoding system, the processor is a core decoder processor and the co-processor is a decoding accelerator adapted to assist the core processor with a decoding function.

22. *(Original)* The device of claim 20 wherein

the processor is a reduced instruction set computer (RISC) processor.

23. *(Cancelled)*

24. *(Cancelled)*

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.